NeuOS: A Latency-Predictable Multi-Dimensional Optimization Framework for DNN-driven Autonomous Systems

Soroush Bateni
The University of Texas at Dallas

Cong Liu
The University of Texas at Dallas
The tale of two worlds

Deep Neural Networks (DNNs)

Autonomous Embedded Systems

Background

Autonomous Decision
The tale of two worlds

Deep Neural Networks (DNNs)

Main Objective
- Maximum Accuracy

Autonomous Embedded Systems

Main Objectives
- Timing predictability
- Energy efficiency
- Safety

Autonomous Decision
Marriage between the two worlds

Deep Neural Networks (DNNs) + Autonomous Embedded Systems

Background
The big picture

Background

Hardware/software stack for executing DNNs in Autonomous Embedded Systems
The focus of related research in AES is currently mostly on the DNN and the hardware.
The big picture

Efficient DNNs
- Quantization
- Lowrank approximation

Background

DNN

↑ Caffe
Framework/OS

JetPack
The big picture

Background

Hardware/software stack for executing DNNs in Autonomous Embedded Systems

Special Processors
- AI accelerators
- DNN-focused SoCs
Where system software/framework can help

Goals

• Meet timing requirements
• Be energy efficient
• Minimize accuracy loss.

All the above goals must be achieved at the same time.
Jack of all trades, master of none

**Timing predictable & energy efficient**
Can be achieved at system level via Dynamic Voltage Frequency Scaling (DVFS).

**Timing predictable & accurate**
Can be achieved at application level via DNN configuration change.

**Master of none**
Combining the two (even at different rates) will yield unpredictable results.
Timing predictable & energy efficient

Can be achieved at system level via Dynamic Voltage Frequency Scaling (DVFS).

Timing predictable & accurate

Can be achieved at application level via DNN configuration change.

Master of none

Combining the two (even at different rates) will yield unpredictable results.

Motivation
Timing predictable & energy efficient

Can be achieved at system level via Dynamic Voltage Frequency Scaling (DVFS).

Timing predictable & accurate

Can be achieved at application level via DNN configuration change.

Master of none

Combining the two (even at different rates) will yield unpredictable results.

Motivation

Jack of all trades, master of none

Need per-layer adjustments.

Need per-layer adjustments.

Need coordination.
No one is alone

**Multiple ResNet-50 instances executed together**

The underlying system-level solution here is PredJoule\(^1\)

![Graph showing latency and energy across different instance IDs](image)

**Takeaways**

The first DNN instance is the winner, other DNN instances not as lucky because the method used here is greedy.

The DVFS configurations chosen only work well for the first DNN instance.

---

No one is alone

**Multiple ResNet-50 instances executed together**

The underlying system-level solution here is PredJoule\(^1\)

![Graph showing latency and energy for different instances]

**Takeaways**

The first DNN instance is the winner, other DNN instances not as lucky because the method used here is greedy. The DVFS configurations chosen only work well for the first DNN instance.

Design Goals

Core Targets

• **Timing predictable**: the system must meet deadlines set by the system designer for the DNN.

• **Energy efficient**: the system must use DVFS to achieve near-optimal energy usage for DNNs.

• **Accurate**: the system can change accuracy dynamically but must do so cautiously.

• **Multi-DNN compatibility**: the system should be able to coordinate and find an efficient solution for all DNN instances.

Optimization Targets

The system must also be flexible to adapt to different system constraints. We offer three optimization targets (switchable by an external policy controller):

• **Min Energy** ($M_p$) is used when our design is deployed in extremely low power scenario such as remote sensing.

• **Max Accuracy** ($M_a$) is used when our design is deployed in extremely mission-critical scenarios.

• **Balanced Energy and Accuracy** is the scenario where our design can choose what is best given the timing requirement.
**Design**

**Timing predictability**

**LAG analysis**

- Keep track of per-layer progress

\[
LAG_i(t, L_i(t)) = \sum_{l \in L_i(t)} (d_l - e_l)
\]

- Accumulative LAG
- Per-layer sub-deadline

**Proportional Deadline**

- Build an ideal schedule by setting sub deadlines in proportion to their execution time

\[
d_l = (e_l / \sum_{x \in L_i} (e_x)) \cdot D_i
\]

- Per-layer sub-deadline
- End-to-end deadline for the DNN instance
Building a cohort

We keep a pair of local variables for each DNN instance.

\[ \Delta \text{ Calculator} \]

1. Based on the last reported values of LAG in the cohort, calculate a speedup (or slowdown)
2. Lookup\(^1\) the best possible DVFS configuration for that slowdown.
3. The output is a list \( \Delta \) of optimal DVFS configurations for each DNN instance.

\[ X_i \text{ Calculator} \]

1. For each element of \( \Delta \), calculate the required (further) speedup (or slowdown) for other DNN instances.
2. This time, lookup\(^1\) the best possible approximation configuration that matches that slowdown.

\(^1\)Please see the paper and the source code for more information.
The decision tree

Cohort

Δ Calculator

Δ = δ₁, δ₂, ..., δₙ

X₁, Calculator

S₁, S₂, ..., Sₙ

Overview of modes

Optimization
Choosing a $\delta$ (DVFS configuration) will have consequences in terms of accuracy for all DNNs in the cohort. Therefore, the question is, which $\delta$ is the best?

- **Min. Energy ($M_P$)** chooses the $\delta$ that has the least PowerUp value in the PowerUp-SpeedUp table, without looking at accuracy loss.

- **Max. Accuracy ($M_A$)** chooses the $\delta$ so to minimize the value of $\sum \delta_i S_{A_i}$.

- **Balanced Energy and Accuracy** uses the Bivariate Regression Analysis (BRA) to achieve a balanced approach backed by statistical analysis of the tree\(^1\).

\(^1\)Please see the paper for more information.
Overview

Based on Caffe

• Available as an open-source project on GitHub
• No need to use APIs
• No need to redesign DNN models
• Need to generate
  • Hash tables
  • Lowrank approximated version of your DNN model.

Tested extensively

• Tested on NVIDIA Jetson TX2 and Jetson AGX Xavier
• Tested using image recognition DNNs
  • AlexNet, GoogleNet, ResNet-50, VGGNet
• Tested using three cohort sizes
  • Small: 1 DNN instance
  • Medium: 2-4 DNN instances
  • Large: 6-8 DNN instances
• We include a mixed scenario that uses a combination of all the DNN models
### Energy Evaluation

**68% avg. improvement on TX2**

**46% avg. improvement on AGX Xavier**

70% avg. improvement on TX2
Energy

Evaluation

<table>
<thead>
<tr>
<th>NeuOS</th>
<th>PredJoule</th>
<th>Poet</th>
<th>Race2Idle</th>
<th>Max-N</th>
<th>Max-Q</th>
</tr>
</thead>
</table>

1 Process

4 Process

8 Process

TX2

AGX

AlexNet | GoogleNet | ResNet-50 | VGGNet

AlexNet | GoogleNet | ResNet-50 | VGGNet | Mixed

AlexNet | GoogleNet | ResNet-50 | VGGNet | Mixed
Evaluation

Latency

<table>
<thead>
<tr>
<th>NeuOS</th>
<th>PredJoule</th>
<th>Poet</th>
<th>Race2Idle</th>
<th>Max-N</th>
<th>Max-Q</th>
</tr>
</thead>
</table>

1 Process:
- TX2
- AGX

4 Process:
- AlexNet
- GoogleNet
- ResNet-50
- VGGRNet

8 Process:
- Mixed

23
**Latency**

**Evaluation**

<table>
<thead>
<tr>
<th></th>
<th>NeuOS</th>
<th>PredJoule</th>
<th>Poet</th>
<th>Race2Idle</th>
<th>Max-N</th>
<th>Max-Q</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Process</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AGX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

68% avg. improvement on TX2
40% avg. improvement on AGX Xavier

53% avg. improvement on TX2
32% avg. improvement on AGX Xavier
**Small cohort**
3.25% deadline miss rate.

<table>
<thead>
<tr>
<th></th>
<th>TX2</th>
<th>AGX Xavier</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>9.2ms</td>
<td>5ms</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>48ms</td>
<td>12ms</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>130.3ms</td>
<td>26.1ms</td>
</tr>
<tr>
<td>VGGNet</td>
<td>39.1ms</td>
<td>36.2ms</td>
</tr>
</tbody>
</table>

**Medium cohort**
Deadline miss rate same as the small cohort.

<table>
<thead>
<tr>
<th></th>
<th>TX2</th>
<th>AGX Xavier</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>10.4ms</td>
<td>11ms</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>39.2ms</td>
<td>12.5ms</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>101.7ms</td>
<td>26.3ms</td>
</tr>
<tr>
<td>VGGNet</td>
<td>69ms</td>
<td>25.9ms</td>
</tr>
</tbody>
</table>

**Large cohort**
Deadline miss rate same as the small cohort.

<table>
<thead>
<tr>
<th></th>
<th>TX2</th>
<th>AGX Xavier</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>13.6ms</td>
<td>10.7ms</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>40.8ms</td>
<td>54ms</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>190ms</td>
<td>62ms</td>
</tr>
<tr>
<td>VGGNet</td>
<td>72ms</td>
<td>36.1ms</td>
</tr>
</tbody>
</table>
Relative Accuracy
Flexibility

(a) The entire configuration space for all DVFS and accuracy combinations for Jetson TX2.

(b) Chosen configurations in the triangle space.
Flexibility

11759 DVFS configurations on Jetson TX2.
51967 DVFS configurations on Jetson AGX Xavier.
Computation

Relatively negligible execution overhead (in ms).

Memory

Overhead includes the lowrank version of each DNN model. The right side shows how much of the total memory of each device is occupied.

<table>
<thead>
<tr>
<th></th>
<th>Overhead in addition to Caffe</th>
<th>Ratio to total memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lowrank</td>
<td>Hash Table</td>
</tr>
<tr>
<td>NeuOS</td>
<td>0.145</td>
<td>0.571</td>
</tr>
<tr>
<td>PredJoule</td>
<td>0.772</td>
<td>0.929</td>
</tr>
<tr>
<td>ApNet</td>
<td>0</td>
<td>3.27</td>
</tr>
<tr>
<td>Poet</td>
<td>151.03</td>
<td>604.12</td>
</tr>
</tbody>
</table>

Evaluation

Overhead
Conclusions

The system community to the rescue

- Certain problems cannot be solved at application level (by AI researchers) and at hardware level separately
  - Ensuring timing predictability, energy efficiency, and accuracy for DNNs in Autonomous Embedded Systems requires coordination

- We presented the design of NeuOS that can achieve these three goals by
  - Using LAG analysis to ensure real-time performance
  - Efficiently propagating all possible choices
  - Having flexibility in terms of choosing the best combination of configurations based on system designer’s criteria or external policy controller

- We extensively evaluated NeuOS
  - Using the latest AES devices
  - Using prominent image recognition DNNs
  - Under multiple configurations, including various cohort sizes
  - Against the most prominent accessible solutions available to researchers.
Thank you

Questions
Please do not hesitate to send your questions to soroush@utdallas.edu.

Source Code
https://github.com/Soroosh129/NeuOS